The *Run Time* metric is a measure of how fast the simulation runs in terms of real clock time, as opposed to simulation time. The run time is a vital issue for analysts using a simulation model. The time it takes to run a model is an important factor that directly affects the ability to deliver timely results based on model analysis. Any improvement in run time (without adversely affecting the simulation behavior) is always desirable.

In the case of the concept models analyzed in this thesis, the run time is a time duration based on Equation 3.1.

$$\text{Run Time} = \text{End Time} - \text{Start Time} - \text{Initialization} \tag{3.1}$$

End Time is the time at which simulation execution ends, **Start Time** is the time at which simulation execution begins, and **Initialization** is the amount of time the simulation spends initializing the model and is characterized by Equation 3.2.

$$\text{Initialization} = \text{Last Entity Arrival Time} - \text{Start Time} \tag{3.2}$$

The models start empty with no entities present. During initialization, the appropriate entities are generated and placed in the proper holding queues in the model (i.e. the original weapon entities are created and loaded into their respective stockpile storage locations). This puts the model in the proper starting state. The arrival of the last initialized entity to its appropriate holding queue (**Last Entity Arrival Time**) signals the end of the initialization period. The initialization period is not included in the run time since initialization takes the same amount of time in each compared case, is very small

(less than 0.5 seconds), and has nothing to do with the impact that the various modeling techniques have on execution speed.

The run time calculation was implemented in code and integrated into the simulation models so that when a simulation run concluded, the run time would automatically be calculated and displayed. Each model case was run five times (five replications) with the run time being recorded for each replication. The sample mean ($\overline{\mathbf{X}}$) and standard deviation (**s**) of the run times were calculated based on Equations 3.3 and 3.4 respectively:

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{3.3}$$

$$s = \sqrt{\frac{\sum_{i=1}^{n} \left[X_i - \overline{X}\right]^2}{n-1}} \tag{3.4}$$

where $\mathbf{X_i}$ is the run time value observed for each of the individual simulation replications and **n** is the number of replications (sample size). The results are included in chapter 4.

The *Size* metric is a quantitative measure of how much a given modeling construct contributes to the size of the model (in terms of bytes). This value is provided automatically in the simulation software by selecting the desired portion of the model to be measured. The memory consumed (number of bytes) by that portion of the model is then displayed.

In the concept models, the sections specifically relating to accessing sub-level entity information for true defect analysis and weapon selection were selected and sized. This allowed the different modeling techniques to be compared according to size. This comparison was valuable in helping to establish and illustrate which methods were more size efficient. Since the sizes of the given modeling constructs are constant between simulation replications, no statistical quantification on the size metric was performed.

3.6 STEP 6 – 2X2 MODEL IMPROVED (7X7 MODEL)

Taking what was learned from the comparative analyses of the concept models, the 2x2 model was improved and expanded using direct sub-level entity access techniques. The new, improved model is hereafter referred to as the "7x7 model." The 7x7 model does everything the 2x2 model does and more. It performs true defect analysis on the stockpile weapon entities, models several different storage sites with all weapons distributed appropriately among them, and tracks seven weapon types, each with seven constituent subsystems (hence the 7x7 designation). Additionally, the weapon types are no longer segregated into different holding queues at each storage site. All weapons at each site reside in a single holding queue.

The entity structure in the 7x7 model is hierarchical, with all sub-level entities (the weapon subsystems) maintaining their own attributes. True defect analysis is performed in the model using the same *global* implementation of *direct sub-level entity access* applied in the True Defect Analysis Mod-1B model. Sorting and selection of weapons from the multiple storage sites is done remotely using the same *global*

implementation of *direct sub-level entity access* as applied in the Multiple Storage Sites Mod-2 model.

Since the 7x7 model is markedly different than the 2x2 model, the two were not quantitatively compared (since the quantitative metrics used to compare the other models would be meaningless). However, some observations relative to the qualitative metrics were made and will be discussed in chapter 4.

CHAPTER 4

RESULTS AND ANALYSIS

The results supporting the use of *direct sub-level entity access* to improve nuclear stockpile simulation modeling come primarily from the comparative analyses of the concept models described in the previous chapter. This current chapter presents the results and observations from the analyses and discusses additional implications relating to the use of *direct sub-level entity access* in nuclear stockpile models. The discussion will be divided into the following sections:

1) Comparison of the True Defect Analysis models
2) Comparison of the Multiple Storage Sites models
3) Comparison of the 2x2 and 7x7 models
4) Potential disadvantages of implementing *direct sub-level entity access.*

4.1 TRUE DEFECT ANALYSIS MODELS

The True Defect Analysis Baseline-1 concept model was compared against the two corresponding modified cases (Mod-1A and Mod-1B). Both quantitative and qualitative metric comparisons were performed to identify the advantages of using *direct sub-level entity access.*

To provide an ample base for comparative observation and analysis, three scenarios were set up under which all the models were run to explore the behaviors under

different input conditions. All input parameters were the same in each of the three scenarios with the exception of how often true defect analysis was performed. Table 4.1 lists the parameter values that varied between the scenarios.

Table 4.1 Scenario Parameter Values for the True Defect Analysis Models

	Models Perform True Defect Analysis Every ...
Scenario 1	13 weeks
Scenario 2	26 weeks
Scenario 3	52 weeks

The three scenarios represent situations where true defect analysis is conducted quarterly, semi-annually, and annually.

4.1.1 Quantitative Analysis

Table 4.2 lists the average run times (as well as the standard deviations) of the three True Defect Analysis models for each of the scenarios tested. The Baseline-1 model implements the common unbatch/batch technique for doing true defect analysis, while the Mod-1A and Mod-1B models represent *local* and *global* implementations of *direct sub-level entity access* used to accomplish true defect analysis. The values in the table are based on five simulation replications of each model/scenario. As mentioned previously in chapter 3, all the models produced identical model output data within a given scenario. This allowed the modeling methods to be compared as opposed to the output data.

Table 4.2 Run Time of the True Defect Analysis Models

	Baseline-1 Model	**Mod-1A Model**	**Mod-1B Model**
Scenario 1 (Avg. / *Std Dev*)	97.2 seconds *0.10 seconds*	56.9 seconds *0.07 seconds*	9.4 seconds *0.00 seconds*
Scenario 2 (Avg. / *Std Dev*)	50.0 seconds *0.05 seconds*	30.5 seconds *0.05 seconds*	5.8 seconds *0.04 seconds*
Scenario 3 (Avg. / *Std Dev*)	27.2 seconds *0.05 seconds*	16.1 seconds *0.05 seconds*	4.1 seconds *0.00 seconds*

One important thing to note about the run times of the models is that there is very little spread in the observed values for any given case (reflected by the standard deviations). These variations are statistically insignificant for the purposes of comparing run times.

The *local* implementation of *direct sub-level entity access* applied in the Mod-1A model resulted in a run time that was between 1.6 and 1.7 times faster than the baseline case (depending on which scenario was run). The *global* implementation of *direct sub-level entity access* applied in the Mod-1B model produced a run time that was between 6.6 and 10.3 times faster than the baseline case (again, depending on the scenario).

The various scenarios are presented to illustrate the point that different models and modeling conditions will affect – to different degrees – the run time improvements achieved from implementing *direct sub-level entity access*. However, the trend holds that significant run time improvements still result from implementing *direct sub-level entity access* in place of more common modeling methods. The actual amount of improvement is model dependent.

Table 4.3 shows the relative size differences (in terms of memory usage) between the unbatch/batch implementation of doing true defect analysis in the baseline case, and the *local* and *global* direct access implementations employed in the Mod-1A and Mod-1B

cases respectively. Since the size of the constructs remain constant between all runs, no reference to the different scenarios is necessary.

Table 4.3 Size Contributed to Model by each True Defect Analysis Implementation

	Baseline-1 Implementation	**Mod-1A Implementation**	**Mod-1B Implementation**
Size (Memory Usage)	158 KB	148 KB	2 KB

A relatively small improvement in size of the *local* access implementation (in the Mod-1A model) occurs compared with the very large improvement in size of the *global* access implementation (in the Mod-1B model). This happens because a significant part of the true defect analysis in the Baseline-1 and Mod-1A cases involves coordinating and controlling the routing of entities to a designated location where the sub-level information is actually read and compiled. The only difference between the Mod-1A case and the Baseline-1 case (see section 3.4.1) is that the *local* implementation of direct sub-level entity access (Mod-1A) does not require the weapon entities to be unbatched and re-batched to get the appropriate information. The weapon entities still have to move through a designated point, however, to be accessed. The *global* implementation of direct sub-level entity access (Mod-1B), on the other hand, eliminates the need to move or route the weapon entities to perform true defect analysis. The overhead that goes into coordinating the movement of entities and compiling the resultant sub-level information in the Baseline-1 and Mod-1A cases is eliminated by applying *global direct sub-level entity access*. The *local* implementation still provides some improvements and benefits

over the baseline method in terms of both size and speed; however, the *global* implementation has an even greater positive impact.

With model size being an important consideration in computer simulation modeling, the Mod-1A and Mod-1B implementations of *local* and *global* sub-level entity access illustrate the potential for size improvements that can be obtained by applying *direct sub-level entity access* in place of more commonly used methods (as in the baseline case). The size is related to the number and type of programming variables used in (and, indirectly, the amount of) the underlying code required to achieve a given modeling implementation. A benefit of smaller size is the reduced hardware resources needed to run the simulation (size efficiency). Also, a much smaller implementation usually means much less code to execute, resulting in faster run times (speed efficiency). Although there is not an exclusive correlation between size and speed, both are useful measures in determining the efficiency of a particular setup.

Overall, the speed and size improvements shown in the modified True Defect Analysis models can be attributed to their more direct approach of getting entity information. Applying *direct sub-level entity access* eliminates the many extra execution steps that more common methods require to perform the same function.

4.1.2 Qualitative Analysis

The implementations of true defect analysis in the baseline model (using the unbatch/batch technique), the Mod-1A model (using *local* direct access), and the Mod-1B model (using *global* direct access) were analyzed according to the qualitative metrics identified previously. A comparison of the three cases is presented in Table 4.4.

Table 4.4 Qualitative Metric Comparison of Defect Analysis Models

Complexity	**Baseline-1 (using batch/unbatch technique):** Overly complex by requiring each entity to move through a given point in the model with each entity being unbatched to gain access to sub-level attribute information and then re-batched to reestablish the proper entity structure. Complexity increases significantly to account for multiple storage sites or additional weapon and subsystem types.
	Mod-1A (using *local* implementation): No unbatching or batching necessary, though each entity still must be moved through a given place in the model to access sub-level information. Complexity increases significantly to account for multiple storage sites (primarily due to work involved with moving the weapons to the appropriate analysis point). However, accounting for additional weapon and subsystem types is not as complicated as in the baseline case.
	Mod-1B (using *global* implementation): No unbatching or batching necessary. No unnecessary coordination or movement of entities is required to get the sub-level information. Negligible increase in complexity to account for multiple storage sites or additional weapon and subsystem types.
Scalability	**Baseline (using batch/unbatch technique):** Not easily scalable to account for additional stockpile storage sites or additional weapon and subsystem types. Each additional stockpile storage site requires cumbersome modeling logic to coordinate the movement of weapon entities from different sites. Additional logic is also required at the defect analysis point to account for each additional weapon or subsystem type. Significant adverse size and speed impacts result.
	Mod-1A (using *local* implementation): Requires a smaller modeling construct at the defect analysis point to account for additional weapon and subsystem types. However, the *local* implementation of *direct sub-level entity access* still requires the additional modeling logic to coordinate movement of the weapons from each stockpile storage site. Hence, size and speed still increase, but less than in the baseline case.
	Mod-1B (using *global* implementation): Very scalable. No modeling logic needed to coordinate movement of entities to the true defect analysis point (since no entity movement is required). Negligible speed and size impact to account for additional storage sites or weapon and subsystem types.
Flexibility/Functionality	**Baseline (using batch/unbatch technique):** Flexibility is very limited and functionally constrained since all weapon entities in the model must be physically moved and manipulated to do true defect analysis. Care must be taken to schedule the true defect analysis when it will not interfere with or coincide with the other functions in the model requiring weapon movement
	Mod-1A (using *local* implementation): Somewhat limited because this implementation is still constrained by how and when entities actually get to the point of analysis. However, there is more functionality and flexibility (more options) in dealing with attributes once the entities arrive at the analysis point.
	Mod-1B (using *global* implementation): Much more flexible and functional. The analysis is not constrained by movement of entities (since no movement is necessary) and more flexibility exists for selecting weapon types and subsystem types for inclusion in the analysis.

Table 4.4 indicates how well each implementation meets the objectives identified earlier for improving the 2x2 nuclear stockpile life-extension model; the objectives being to perform true defect analysis, account for multiple weapon storage sites, and include more weapon and subsystem types. All three cases were designed specifically to demonstrate alternative approaches to true defect analysis, so they all potentially fulfill the first objective. However, as indicated in the table, the complexity and scalability issues involved with accounting for multiple storage sites and additional weapon and subsystem types make the Baseline-1 and Mod-1A implementations of true defect analysis much less desirable that the *global* direct access implementation in Mod-1B. With the *global* approach, all three objectives can be appropriately addressed without the true defect analysis portion of the model becoming unduly complicated or creating an adverse impact on the rest of the model.

4.2 MULTIPLE STORAGE SITES MODELS

The Multiple Storage Sites Baseline-2 concept model was compared against the corresponding modified case (Mod-2). Both quantitative and qualitative comparisons were performed according to the previously defined metrics to identify the advantages of using *direct sub-level entity access.*

Similar to what was done previously in section 4.1, three scenarios were set up under which the two Multiple Storage Sites models were run to provide an ample base for observation and analysis under different input conditions. All input parameters were the same in each of the three scenarios with the exception of how often weapon entities had

to be selected from the multiple storage sites for maintenance. Table 4.5 lists the parameter values that varied between the scenarios.

Table 4.5 Scenario Parameter Values for the Multiple Storage Sites Models

	Schedule Maintenance on a Set of Weapon Entities Every ...
Scenario 1	13 weeks
Scenario 2	26 weeks
Scenario 3	52 weeks

The three scenarios represent situations where scheduled maintenance requests require the selection of weapon entities from multiple storage sites quarterly, semi-annually, and annually.

4.2.1 Quantitative Analysis

Table 4.6 lists the average run times (as well as the standard deviation) of the two Multiple Storage Sites models for each of the scenarios tested. The Baseline-2 model implements a common attribute-copying technique for making sub-level entity information available during the selection of entities from the multiple storage sites. The Mod-1 model represents a *global* implementation of *direct sub-level entity access* used to accomplish weapon entity selection. The values in the table are based on five simulation replications of each model/scenario. Again, all the models produced identical model output data within a given scenario allowing the modeling methods to be compared as opposed to the output data.

Table 4.6 Run Time of the Multiple Storage Sites Models

	Baseline-2 Model	**Mod-2 Model**
Scenario 1 (Avg. / *Std Dev*)	36.1 seconds *0.07 seconds*	5.3 seconds *0.00 seconds*
Scenario 2 (Avg. / *Std Dev*)	72.1 seconds *0.11 seconds*	10.6 seconds *0.00 seconds*
Scenario 3 (Avg. / *Std Dev*)	138.6 seconds *0.10 seconds*	20.4 seconds *0.05 seconds*

As was the case with the True Defect Analysis models, the variations in run times for the Multiple Storage Sites models are statistically insignificant for the purposes of the comparison.

The *global* implementation of *direct sub-level entity access* applied in the Mod-2 model produced a run time that was 6.8 times faster than the baseline case for all the scenarios. The run time improvement in this case was essentially constant across the scenarios because the only functions being performed in the models related directly to the selection and scheduling of weapon entities (the focus of the varied parameter) with no other operations occurring in the model. The situation was somewhat different with the True Defect Analysis models in section 4.1.1. In those models, more operations (both true defect and surveillance defect testing) were present, while only one (true defect analysis scheduling) was being varied, resulting in a range of run time improvements across the scenarios.

The run time observations made here (as well as in section 4.1.1) again illustrate the potential speed benefits of applying *direct sub-level entity access* in place of more common modeling methods while realizing that the actual amount of improvement is still model dependent.

Table 4.7 shows the relative size difference (in terms of memory usage) between the common attribute-copying implementation of selecting and sorting weapon entities from multiple storage sites (the Baseline-2 case), and the remote, *global* direct access implementation employed in the Mod-2 case. Since the size of the different constructs remain constant between all runs, no reference to the different scenarios is necessary.

Table 4.7 Size Contributed to Model by each Sort and Select Implementation

	Baseline-2 Implementation	**Mod-2 Implementation**
Size (Memory Usage)	170 KB	31 KB

As was the case with the *global* true defect analysis implementation, the Mod-2 implementation of remote, *global* sub-level entity access illustrates the potential for significant size reductions that can be obtained by applying *direct sub-level entity access* in place of more commonly used methods.

The speed and size improvements shown in the modified Multiple Storage Sites model can largely be attributed to its direct and efficient approach of accessing and using sub-level entity information. Applying *direct sub-level entity access* in this case eliminates – as in other direct sub-level entity access implementations – the many extra execution steps that more common methods require to perform the same function.

4.2.2 Qualitative Analysis

The implementations of sorting and selecting entities from multiple storage sites in the Baseline-2 model (using attribute-copying to access sub-level information) and the Mod-2 model (using remote, *global* direct access) were analyzed qualitatively. A

comparison of the two cases – in terms of the qualitative metrics complexity, scalability, and flexibility/functionality – is presented in Table 4.8.

Table 4.8 Qualitative Metric Comparison of Multiple Storage Sites Models

Complexity	**Baseline-2 (using attribute-copying):** Overly complex by requiring all entities of a requested type to move from all storage sites to a given location in the model to sort and select the appropriate ones to send on. Complexity increases significantly to account for additional weapon and subsystem types.
	Mod-2 (using *global* implementation): No movement of entities is required to sort and select the weapons. Sorting and selection are done globally and remotely, independent of where the weapon entities are in the model and independent of the number of storage sites involved. The complexity remains essentially constant even when more weapon and subsystem types are added.
Scalability	**Baseline-2 (using attribute-copying):** Not easily scalable to account for additional stockpile storage sites or additional weapon and subsystem types. Each additional stockpile storage site and weapon and subsystem type requires additional cumbersome modeling logic to coordinate the movement of the appropriate weapon entities from the different sites. Each storage site must be augmented with additional holding queues and routing logic to keep additional weapon types segregated.
	Mod-2 (using *global* implementation): Very scalable. No modeling logic is needed to coordinate movement of entities since no movement of entities is necessary to sort and select. No segregation of weapon entity types is required. All weapon entities can be kept in a single holding queue at a given storage site (independent of the number of weapon types).
Flexibility/Functionality	**Baseline-2 (using attribute-copying):** Flexibility is very limited and functionally constrained in part because all weapon entities in the model must be physically moved and manipulated to sort and select the appropriate weapon entities. Furthermore, the common modeling constructs used are limited in the number of criteria considered and the ability to sort based only on top-level entity attributes.
	Mod-2 (using *global* implementation): Much more flexible and functional since sorting and selection are unconstrained by the movement of entities (no movement is necessary). Many more sorting and selection options are available, including the ability to select entities based on both top-level and sub-level attribute information simultaneously.

Table 4.8 indicates how well each implementation meets the objectives identified earlier for improving the 2x2 nuclear stockpile, life-extension model; the objectives being to perform true defect analysis, account for multiple weapon storage sites, and include

more weapon and subsystem types. Both of the models were designed specifically to demonstrate how selection of specific weapon entities (in response to a surveillance or maintenance request) can take place in a multiple storage site environment, so both potentially fulfill the second objective. However, as alluded to in the table, the complexity, scalability, and functionality issues involved with accounting for multiple storage sites and additional weapon and subsystem types make the Baseline-2 implementation of sorting/selection much less desirable than the *global* direct access implementation of the Mod-2 model. Using the *global* approach, the two objectives relating to multiple storage sites and increased weapon and subsystem types can be appropriately addressed without contributing adversely to or being an undue burden on any other part of the model (including true defect analysis).

4.3 ORIGINAL 2X2 MODEL VS. IMPROVED 7X7 MODEL

One of the goals of the concept models was to determine the best ways to improve the original 2x2 nuclear stockpile model relative to the key deficiencies originally identified. The analyses of the various implementations in the concept models led to applying *direct sub-level entity access* to upgrade the 2x2 model – resulting in the improved 7x7 model.

Applying the same quantitative metrics to a comparison of the 2x2 and 7x7 models (as was done with the concept models) is not valid because of the differences between the 2x2 and 7x7. However, the improvements made possible by *direct sub-level entity access* as implemented in the 7x7 model can be qualitatively contrasted with the 2x2 model. Table 4.9 compares the two models on the bases of the qualitative metrics

complexity, scalability, and flexibility/functionality – highlighting the deficiencies previously identified in the 2x2 model and the ability of the 7x7 model to address those deficiencies.

Table 4.9 Qualitative Metric Comparison of the 2x2 versus the 7x7 Model

Complexity	**2x2 Model:** The simplifying assumptions made in the model are intended to reduce the complexity for the case of two weapon types and two subsystem types. However, the complexity increases significantly if the common modeling constructs and methods used in the 2x2 are merely extended to account for more weapons, subsystems, and storage sites.
	7x7 Model: Fundamentally no more complex than the 2x2 model – and in many regards, less complex. "Hard wired" components of scheduling and routing in the 2x2 model are handled more gracefully and less intrusively in the 7x7 model.
Scalability	**2x2 Model:** Not easily scalable within the existing modeling constructs to account for additional stockpile storage sites or additional weapon and subsystem types. An increase in weapon types, subsystem types, and storage sites alone significantly increases the burden on the model even before being able to address all the functional deficiencies (like true defect analysis).
	7x7 Model: Very scalable with the application of direct sub-level entity access techniques. The model can be further scaled with additional storage sites and more weapon and subsystem types without adversely impacting the key objectives identified earlier (true defect analysis, multiple storage sites, more weapon and subsystem types). The *direct sub-level entity access* implementations are more easily expandable (with less loss of efficiency and function) than the common modeling techniques used in the 2x2 and baseline concept models.
Flexibility/Functionality	**2x2 Model:** Flexibility is very limited. With most functions "hard-wired" using common modeling methods, changes and improvements are more difficult and tedious to make if done within the existing architecture using the same common modeling methods. The functionality of the model is limited to that supported by the common modeling constructs (limited sorting capabilities, only top-level entity manipulation, routing constraints, etc.). In its present state, the model is obviously deficient in defect analysis capabilities and in its ability to handle more storage sites and more weapon and subsystem types.
	7x7 Model: Addresses all the key deficiencies from the 2x2 model with the flexibility to expand and change the model more gracefully and robustly. Functionally more capable, enabling more flexible and powerful selection criteria for maintenance and surveillance requests, and offering more control over defect analysis information.

The 2x2 model and the baseline concept models all employ similar common modeling techniques to perform their functions. Likewise, the 7x7 model and the modified concept models all employ direct sub-level entity access techniques to perform certain key functions. So analogous qualitative observations can be made between the 2x2 and 7x7 models as were made between the baseline and modified concept models. Consequently, Table 4.9 does not attempt to list all the similar observations that can be made based on the observations in Tables 4.4 and 4.8. Also, the observations made in Table 4.9 are primarily focused on those functions relating to the key deficiencies/objectives identified for the 2x2 model.

The observations made between the 2x2 model and the 7x7 model help demonstrate the improvements and impact that *direct sub-level entity access* can have when seeking more flexible, functional, useful stockpile life-extension models. All the advantages observed from applying *direct sub-level entity access* over more common modeling techniques in the concept models are combined in the 7x7 model and result in a more informative, efficient, functional, and robust model than could have been achieved using the more common modeling methods. The 7x7 model performs true defect analysis, enabling more accurate and insightful surveillance program analysis (by being able to compare surveillance data to expected actual defect rates). The 7x7 model can more easily handle the complexities of routing, selecting, sorting, etc. that accompany a stockpile consisting of many weapon and subsystem types. The hierarchical weapon entity structure represented and supported in the 7x7 model more closely depicts the actual situation and lets the model more effectively communicate information to model users and developers. The implementations of *direct sub-level entity access* in the 7x7

model enhance all of these functions – resulting in a model that is more easily expandable and flexible to change, making the model even more useful for the future.

4.4 DISADVANTAGES OF DIRECT SUB-LEVEL ENTITY ACCESS

Several advantages of using *direct sub-level entity access* have been addressed in the previous sections. Applying the concept resulted in smaller, faster, more efficient, and more flexible model implementations. However, a few potential disadvantages of *direct sub-level entity access* should be mentioned.

The primary disadvantage of *direct sub-level entity access* (as experienced in this thesis) is the initial effort required to enable the capability in a given simulation software product. This is because *direct sub-level entity access* is not currently a very common modeling technique. Most discrete-event simulation software products do not already have the built-in capability for direct sub-level entity access. That is, the functionality is not already a pre-defined function readily available to the user. Depending on the software being used, enabling direct sub-level entity access requires a significant amount of initial effort – including a substantial amount of programming to get things set up properly. For example, the special functional blocks (see Appendices B, C, and D) that were used in the modified concept models required substantial custom programming to interface with the existing sub-level entity model data properly. On the other hand, a few simulation software products do support some degree of sub-level entity access more readily than others. In such cases, the initial effort that is required to enable *direct sub-level entity access* in the desired way may be significantly reduced.

If *direct sub-level entity access* is not a built-in function of a given simulation product, software maintenance and support for the functionality (as built-in by the user) is left to the user. Unless it is a built-in function of a simulation software product, the user must undertake many of the technical support and maintenance issues relating to the direct sub-level entity access capability.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

The purpose of this thesis was to demonstrate that *direct sub-level entity access* is indeed a useful concept that is often preferable in modeling nuclear stockpile life-extension issues. *Direct sub-level entity access* – a seldom-used concept – had not previously been applied to nuclear stockpile models. However, the application of the concept to such models in this thesis showed that key functions of nuclear stockpile, life-extension models are well suited to, and can benefit greatly from, the advantages *direct sub-level entity access* offers over more common modeling techniques.

Concept models that were designed to address key deficiencies identified in current stockpile life-extension modeling efforts embodied in the 2x2 model were used to help analyze the effectiveness of common modeling techniques compared with direct sub-level entity access techniques. The baseline concept models employed common modeling techniques, including unbatch/batch and attribute-copying, to access and use sub-level entity information from weapon subsystems. The modified concept models used direct sub-level entity access methods to access sub-level entity information. In all cases, the direct sub-level access approach outperformed the more common approaches. *Direct sub-level entity access*, in these cases, resulted in models that were significantly faster, smaller, more efficient, more flexible to change, more scalable, and more capable

than their baseline counterparts. The analysis of the concept model implementations indicated that the best way to meet the objectives of improving the deficient 2x2 nuclear stockpile, life-extension model was to apply global, *direct sub-level entity access*.

Applying *direct sub-level entity access* to the 7x7 model completely addressed the key deficiencies identified from the 2x2 model; namely true defect analysis, multiple storage sites, and additional weapon and subsystem types. Had common modeling techniques been used to expand the 2x2 model (as was demonstrated in part in the baseline concept models), the resulting limitations would have made improving the 2x2 model very difficult and even impractical. Such limitations include cumbersome routing constraints, restrictive entity sorting capabilities, inefficient entity information access, and excessive model size growth; all leading to a slow, inflexible, and overly complicated model. Instead, the 7x7 model's use of *direct sub-level entity access* resulted in a model that is more capable, flexible, scalable, and informative than the original 2x2 model. The 7x7 model is also faster, more efficient, and better poised for future growth than it would have otherwise been had common modeling techniques been used.

The 7x7 model embodies some of the real complexities involved with the nuclear stockpile. Many different weapons and subsystems spread amongst many different storage sites can make modeling the system quite difficult. However, *direct sub-level entity access* (particularly *global* access) was shown to be well suited to handle such complex representations and proved critical to achieving the objectives of the improved stockpile life-extension model.

Some additional observations should be reiterated at this point relating to *direct sub-level entity access*. The concept, as presented in this thesis, is intended to make the

most use of model information as it already exists in the simulation software. The intent is not for users to create or set up their own custom databases (or something similar) to hold model information – essentially making a redundant set of data that the simulation software already manages. Instead, *direct sub-level entity access* strives to take advantage of the fact that the software already keeps track of sub-level entity information, and that this information can and should be more accessible to the modeler – without requiring that everything be done at the top entity level. Consequently, the concept may require a greater initial investment of effort to interface appropriately with the existing sub-level entity model data. The amount of effort required depends largely on the simulation software being used and its existing capabilities relating to sub-level entity access. However, considering the advantages that *direct sub-level entity access* provides, the time and effort spent to make it work in a model is worth it.

5.2 RECOMMENDATIONS

Some recommendations for further research that relate to the thesis include deeper sub-level implementation of direct entity access, database integration, and additional applications for *direct sub-level entity access*. These ideas will be briefly described below.

Using *direct sub-level entity access* to improve nuclear stockpile, life-extension models was the primary focus of this thesis. However, the concept is not restricted to this area of application and would be well suited to a myriad of other interesting simulation modeling areas outside of the nuclear weapon domain. Many things can be and are modeled as hierarchical entity structures (for example, automobiles). Any model with

such entity structures could potentially benefit from *direct sub-level entity access* (depending on the objectives of the model). Investigating different areas of application would be worthy of further consideration.

The specific implementations of *direct sub-level entity access* in this thesis only dealt with two-level entity structures (as in Figure 1.1). The special functional blocks that were created to apply *direct sub-level entity access* in the modified concept models were only designed to support access to information from model entities existing at the first sub-level of a given entity hierarchy. This is because the entity structures in the models analyzed herein were not composed of more than two levels. Supporting *direct sub-level entity access* of deeper sub-levels (as in Figure 1.2) would likely require significantly more effort to apply. Investigating the impact of deeper *direct sub-level entity access* and implementing support for it would be a good candidate for further research. Such an effort could be applied to nuclear stockpile models as was done in this thesis, or to any other appropriate area of application.

Setting up a custom database and interface that tracks sub-level entity model data was mentioned previously as another seldom-used alternative for accessing and using entity information. While this option was judged to require more effort to implement than doing single-depth *direct sub-level entity access*, it would still be interesting to investigate various database options that could be more fully integrated into simulation models from any application domain (weapon or otherwise).

APPENDIX A

VENDOR QUESTIONNAIRE

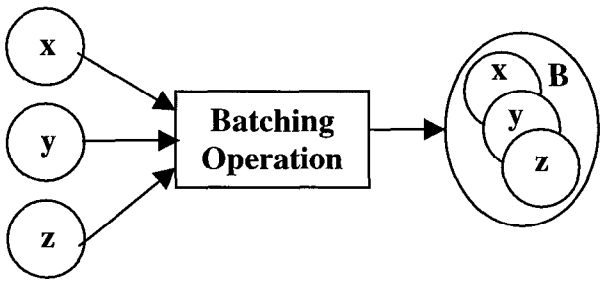
Batched (Sub-Level) Entity Access Questionnaire

Thank you for taking the time to complete this questionnaire. Feel free to qualify any responses or to attach/include additional explanations as you deem necessary.

Background Information/Example:
Consider a simulation model representing three entities (x, y, and z) each with unique characteristics (attributes). For example, each entity carries the *attributes* "type" and "age" and "changeFlag", but with different attribute values:

	"type" value	"age" value	"changeFlag" value
Entity x	0	12	0
Entity y	1	8	0
Entity z	2	17	0

The three entities are batched together in an assembly-type operation. The resulting assembly is a new entity 'B'. Because the assembly may at some future time be separated into its original constituencies, entities x, y, and z must maintain their unique characteristics while batched together. Entity B (the batch) may even have unique attributes assigned to it (such as "AssemblyDate" and "type", etc.). The process being modeled is illustrated below.



At this point entity B (the assembly) is considered a ***top-level*** entity. Entities x, y, and z are considered ***sub-level*** entities (constituent entities of a batch).

As you proceed with the questionnaire, please keep in mind the above example and how it would be implemented using your simulation software. The questions refer primarily to access and control of sub-level entity information (i.e. the attributes on x, y, and z) while in the *batched* state. This means accessing/controlling the sub-level entity

attributes without unbatching the assembly to gain access to them, or without copying the sub-level information onto the top-level entity where it might be more easily accessible. (***Access to*** the attributes implies the ability to read/audit the attribute information. ***Control of*** the attributes implies the ability to change and add/remove attribute information.)

Questions:

1. **Please name the discrete-event simulation product, including version number, for which information is being provided in this questionnaire:**

2. **Is the primary interface to modeling with your product (for the typical user):**
   2a) Programming (writing code)?
   ☐ YES
   ☐ NO

   2b) Or is it primarily graphically based (icon/menu/dialog based, point/click, etc.)?
   ☐ YES
   ☐ NO

3. **Is the typical user of your product usually required to program (write code) in order to build substantial models?**
   ☐ YES
   ☐ NO

4. **How would you characterize your product: (mark all that apply)**
   ☐ Simulator
   ☐ Simulation Language
   ☐ Other (please specify)

5. **Assume a model builder with no programming experience. Would s/he be a typical user of your product?**
   ☐ YES
   ☐ NO

6. **In a model built with your product, do individual entities carry their own attributes?**
   ☐ YES
   ☐ NO (If NO, please explain how entity specific characteristics are tracked)

7. **Does your product offer the capability to batch entities AND preserve their unique attributes, such that when the batch is subsequently unbatched, the original entities and attributes can be restored?**
   - ☐ YES
   - ☐ NO

8. **In a model where entities are batched together (as in the background example) are the attributes on sub-level entities (x, y, and z in the example) accessible *while* they are part of a batch (batch entity B in the example)?**
   - ☐ YES
   - ☐ NO

   If YES:

   8a) Can the sub-level entity's attributes be read/viewed while still part of the batch?
   - ☐ YES
   - ☐ NO

   8b) Can the sub-level entity's attributes be changed while still part of the batch?
   - ☐ YES
   - ☐ NO

   8c) Is sub-level entity attribute access supported at multiple levels of batching? (this means access to a sub-level entity that is part of a batch that, in turn, is also part of another batch ...)
   - ☐ YES
   - ☐ NO

9. **If sub-level entity access IS possible (as described in questions 8 – 8b):**

   9a) Is the capability a built-in functionality that could be used by a "non-programming" user, (i.e. a pre-defined, built-in function/module or capability that does not require custom "programming" to achieve)?
   - ☐ YES
   - ☐ NO

   9b) If the capability is "built-in" (i.e. if you answered YES on question 9a) and does not require programming to achieve, please list the function/module/etc. that performs this action?

   9c) If the capability would require programming to achieve the functionality, how much programming would be required (on a scale of 1 to 5)?

Very Little				Extensive
1	2	3	4	5

10. If sub-level entity attribute access/control IS indeed possible:

10a) Could this access be invoked from a single place in the model to act on entities at other places in the model (i.e. global control)?

☐ YES
☐ NO

10b) Or would the batch entities have to be passed/moved through specific locations/activities in the model in order to have their sub-entities accessed?

☐ YES
☐ NO

11. Do you think users would benefit, or see as useful, built-in capabilities to allow sub-level entity access/control (as it has been discussed in this questionnaire)?

☐ YES
☐ NO

12. A couple of ways to have some degree of access to sub-level attribute information without directly accessing the sub-level entities while in the batched state were alluded to in the background example. That is, one could unbatch the batched entity (essentially bringing the sub-level entities to the top level again), manipulate the attributes as desired, and then batch the entities back together again. Another way would be to copy the sub-level entity attributes onto the top-level batch during the batching operation such that the information would be available at the top-level. Aside from these workarounds, is there another straightforward way to achieve the same goal using your product that has not already been addressed in this questionnaire?

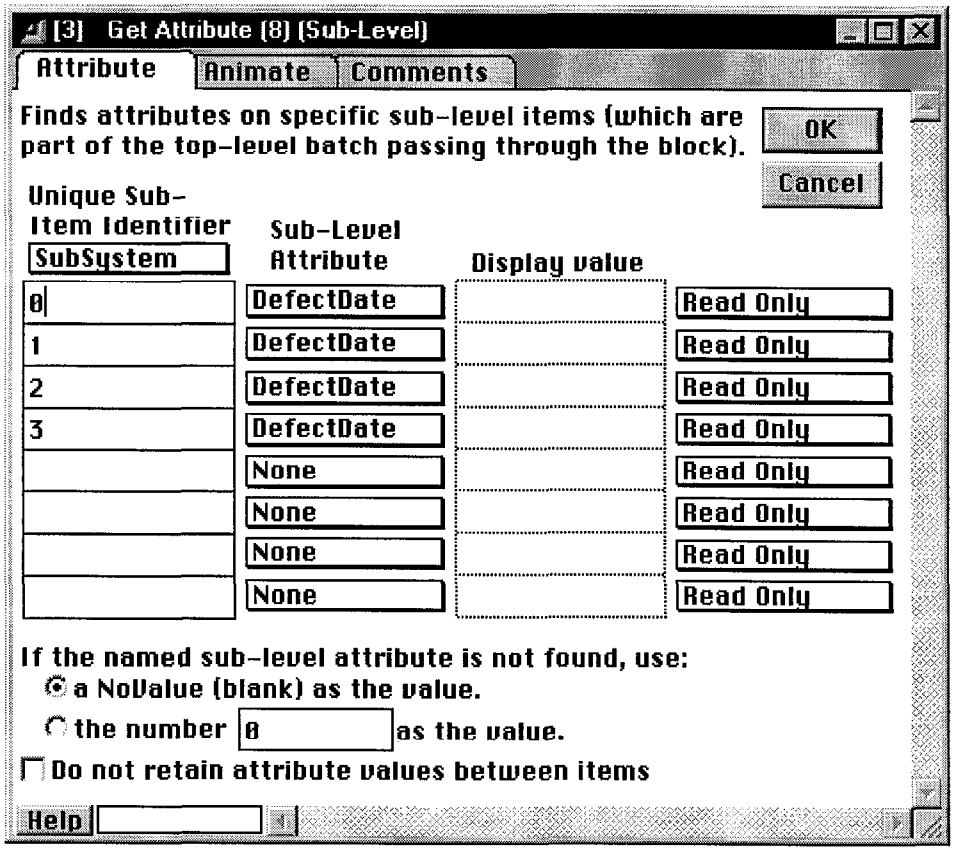
☐ YES (If YES, please explain)
☐ NO

13. Please list any other information that you think would be relevant, or any other related work (published papers, white papers, etc.) that you are aware of pertaining to the topic of sub-level entity access/control.

APPENDIX B

DIALOG OF GET ATTRIBUTE (SUB-LEVEL) BLOCK

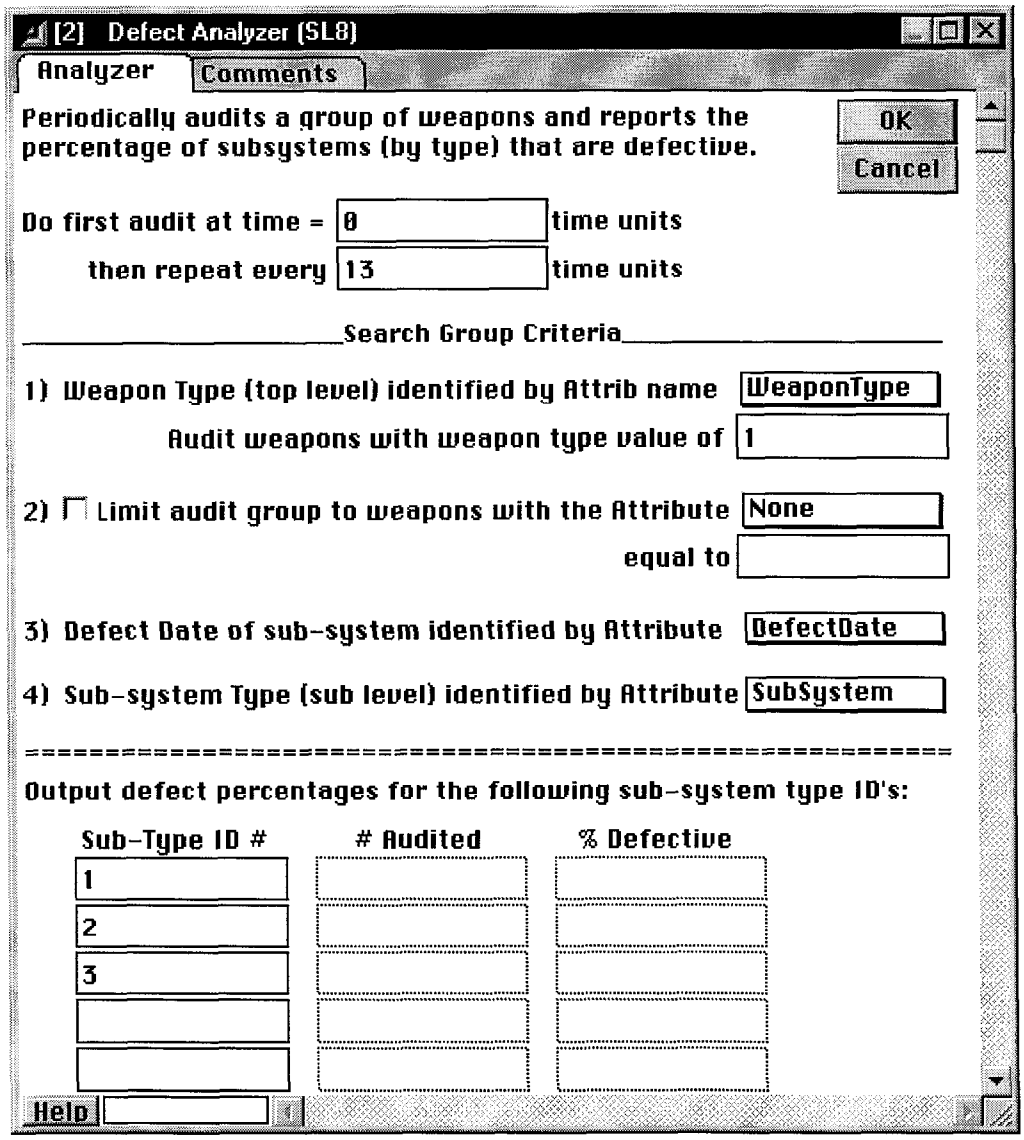
This appendix shows the dialog (the primary user interface) of the *Get Attribute (sub-level)* custom Extend block. Only relevant dialog tabs are shown.



APPENDIX C

DIALOG OF DEFECT ANALYZER BLOCK

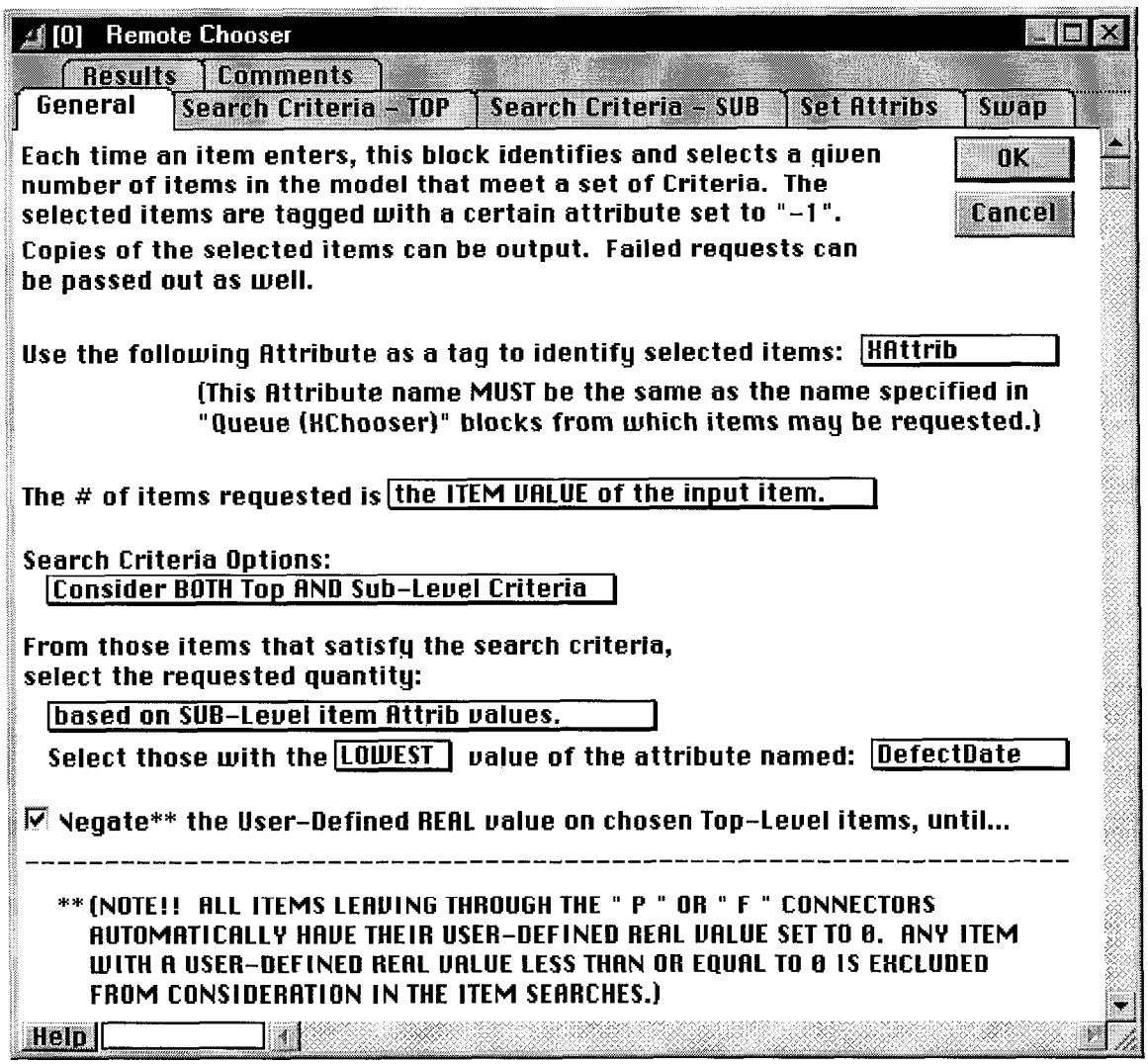
This appendix shows the dialog (the primary user interface) of the *Defect Analyzer* custom Extend block. Only relevant dialog tabs are shown.

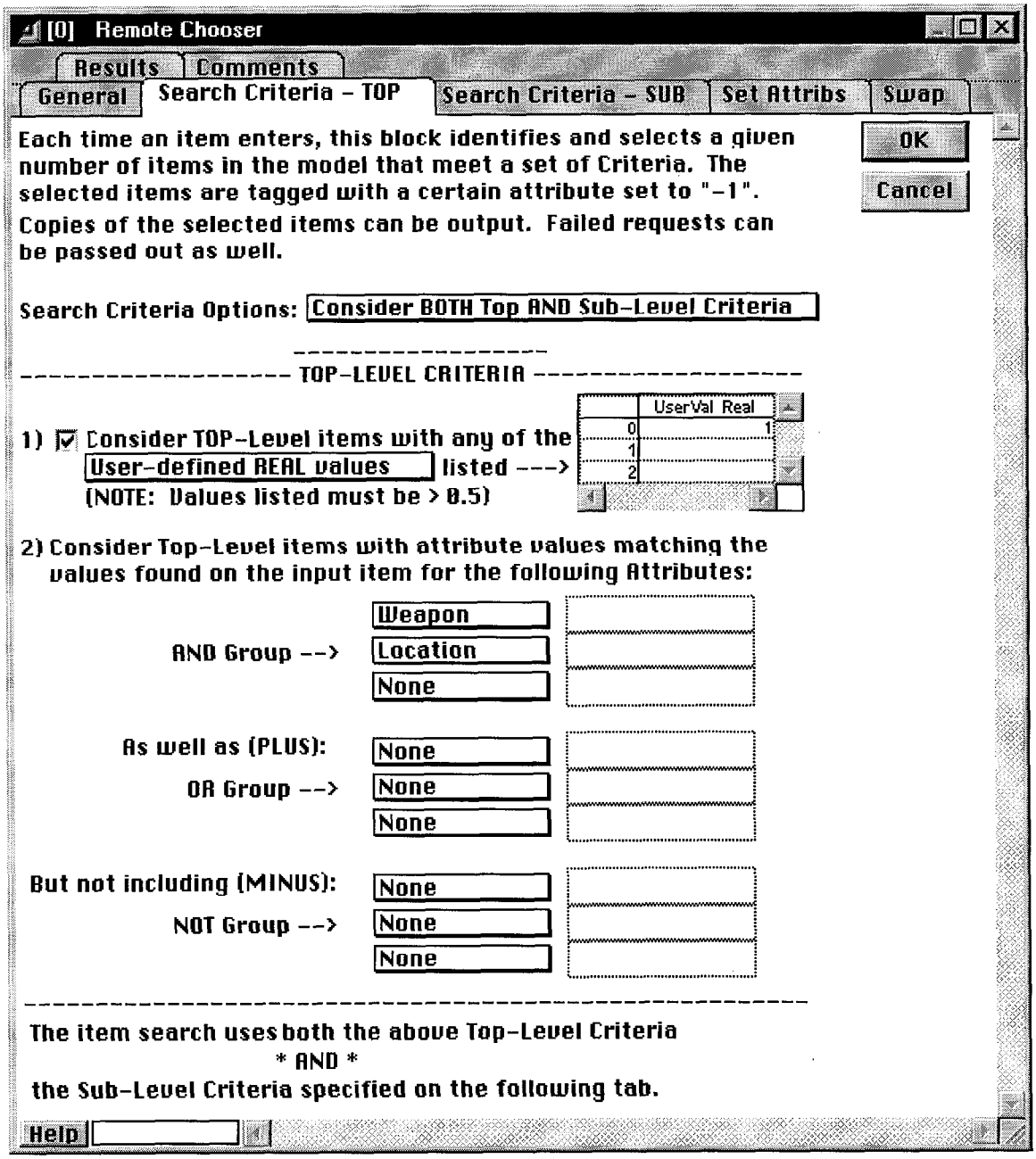


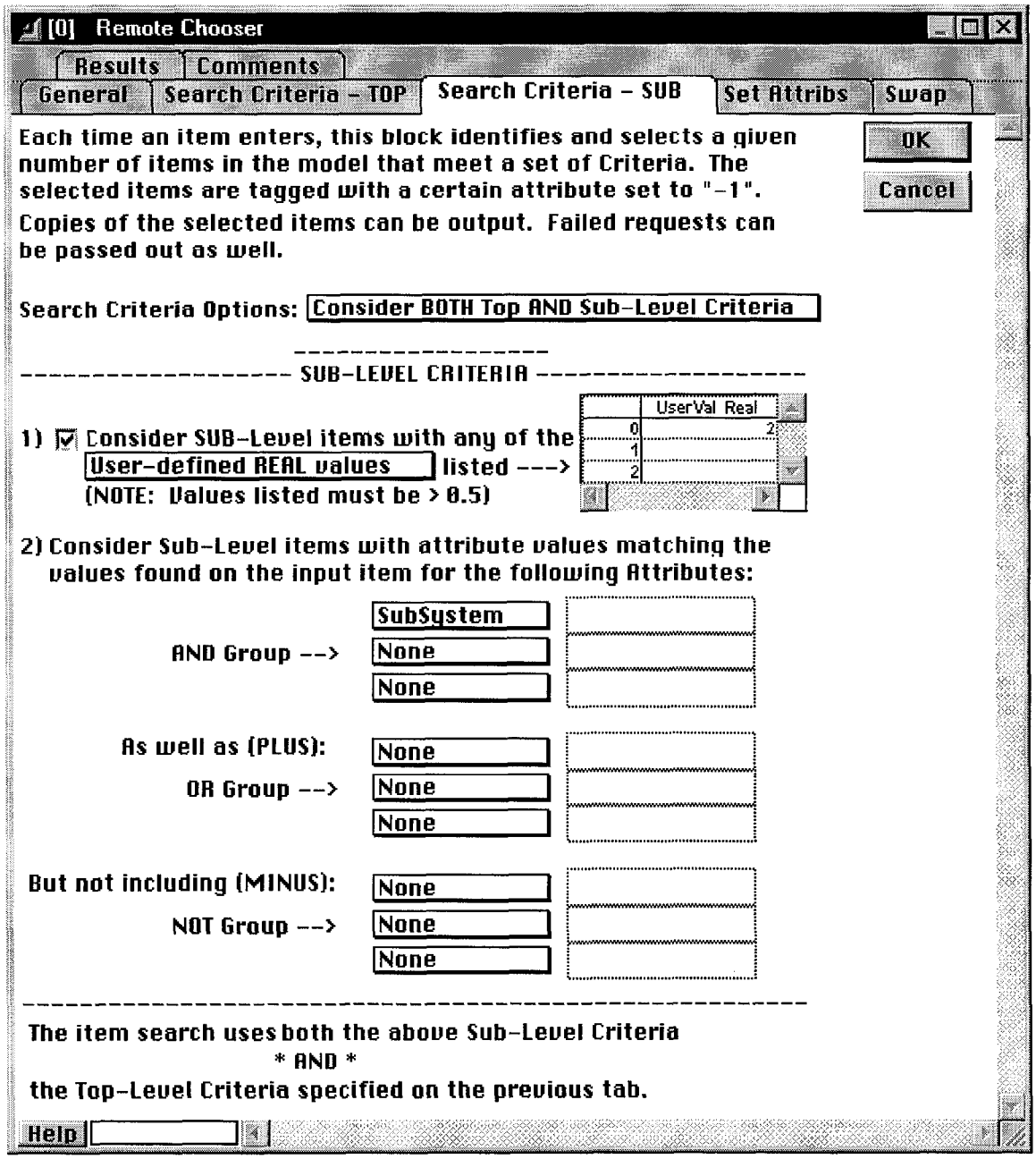
APPENDIX D

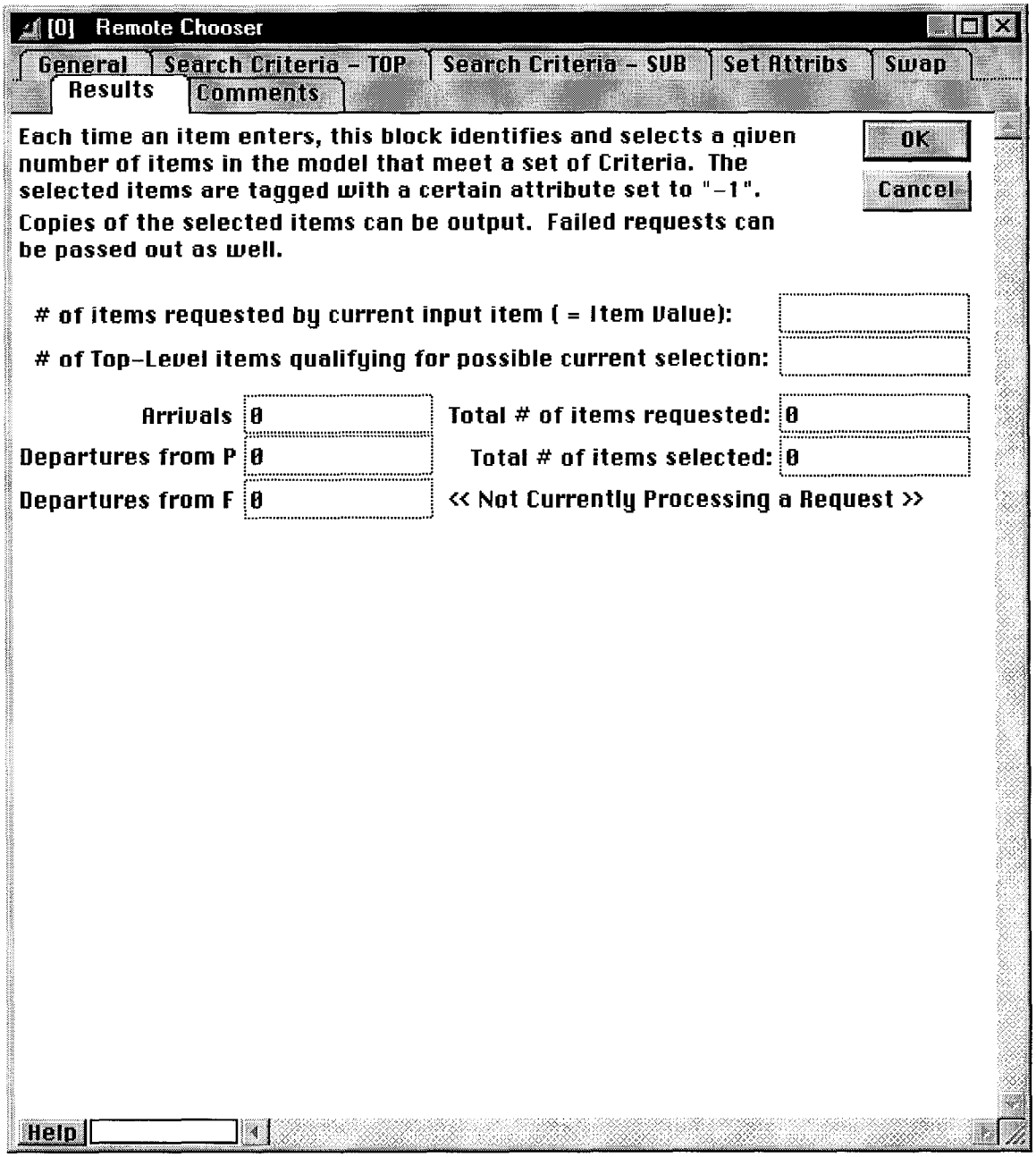
DIALOG OF REMOTE CHOOSER BLOCK

This appendix shows the dialog (the primary user interface) of the *Remote Chooser* custom Extend block. Only relevant dialog tabs are shown.




[0] Remote Chooser
Results
Comments
General
Search Criteria - TOP
Search Criteria - SUB
Set Attribs
Swap
Each time an item enters, this block identifies and selects a given number of items in the model that meet a set of Criteria. The selected items are tagged with a certain attribute set to "-1".
Copies of the selected items can be output. Failed requests can be passed out as well.
OK
Cancel
Search Criteria Options: Consider BOTH Top AND Sub-Level Criteria
TOP-LEVEL CRITERIA
1) Consider TOP-Level items with any of the User-defined REAL values listed --->
(NOTE: Values listed must be > 0.5)
UserVal Real
0
1
2
2) Consider Top-Level items with attribute values matching the values found on the input item for the following Attributes:
AND Group -->
Weapon
Location
None
As well as (PLUS):
OR Group -->
None
None
None
But not including (MINUS):
NOT Group -->
None
None
None
The item search uses both the above Top-Level Criteria
* AND *
the Sub-Level Criteria specified on the following tab.
Help


[0] Remote Chooser
Results
Comments
General
Search Criteria - TOP
Search Criteria - SUB
Set Attribs
Swap
OK
Cancel
Each time an item enters, this block identifies and selects a given number of items in the model that meet a set of Criteria. The selected items are tagged with a certain attribute set to "-1".
Copies of the selected items can be output. Failed requests can be passed out as well.
Search Criteria Options: Consider BOTH Top AND Sub-Level Criteria
SUB-LEVEL CRITERIA
1) Consider SUB-Level items with any of the User-defined REAL values listed --->
(NOTE: Values listed must be > 0.5)
UserVal Real
0
1
2
2) Consider Sub-Level items with attribute values matching the values found on the input item for the following Attributes:
AND Group -->
SubSystem
None
None
As well as (PLUS):
OR Group -->
None
None
None
But not including (MINUS):
NOT Group -->
None
None
None
The item search uses both the above Sub-Level Criteria
* AND *
the Top-Level Criteria specified on the previous tab.
Help


[0] Remote Chooser
General
Search Criteria - TOP
Search Criteria - SUB
Set Attribs
Swap
Results
Comments
Each time an item enters, this block identifies and selects a given number of items in the model that meet a set of Criteria. The selected items are tagged with a certain attribute set to "-1".
Copies of the selected items can be output. Failed requests can be passed out as well.
OK
Cancel
# of items requested by current input item (= Item Value):
# of Top-Level items qualifying for possible current selection:
Arrivals
0
Total # of items requested:
0
Departures from P
0
Total # of items selected:
0
Departures from F
0
<< Not Currently Processing a Request >>
Help

REFERENCES

"1997 Simulation Software Survey." OR/MS Today Oct. 1997. 26 Jan. 1999 <http://lionhrtpub.com/orms/surveys/Simulation/Simulation.html>.

"1998 Simulation Software Survey." APICS - The Performance Advantage July 1998: 56-59.

"APICS - The Performance Advantage 1997 Simulation Software Survey." APICS - The Performance Advantage July 1997. 5 Feb. 1998 <http://lionhrtpub.com/apics/surveys/Simulation/APICS-Simulation.html>.

Balci, Osman. Questionnaire response for Visual Simulation Environment. 23 June 1998.

Balci, Osman, et al. "Dynamic Object Decomposition in the Visual Simulation Environment." Proceedings of the 11th European Simulation Multiconference. Ed. Ali. R. Kaylan and Axel Lehman. San Diego: Society for Computer Simulation, 1997. 69-73.

Balci, Osman, et al. "Visual Simulation Environment" Proceedings of the 1998 Winter Simulation Conference. Ed. D. J. Medeiros, Edward F. Watson, John S. Carson, and Mani S. Manivannan. Vol. 1. San Diego: Society for Computer Simulation, 1998. 279-287.

Banks, Jerry, John S. Carson, II, and Barry L. Nelson. Discrete-Event System Simulation. Upper Saddle River: Prentice Hall, 1996.

Barnes, Catherine. Questionnaire response for Micro Saint. 16 June 1998.

Barnes, Martin. Questionnaire response for QUEST. 26 June 1998.

Boerigter, Stephen. Personal Interview. 22 Jan. 1999.

Centeno, Martha A., and M. Florencia Reyes. "So You Have Your Model: What To Do Next – A Tutorial on Simulation Output Analysis." Proceedings of the 1998 Winter Simulation Conference. Ed. D. J. Medeiros, Edward F. Watson, John S. Carson, and Mani S. Manivannan. Vol. 1. San Diego: Society for Computer Simulation, 1998. 23-29.

Demuth, Nelson, Stephen Nielson, and Stephen Boerigter. "Hierarchical Recursive Modeling Approach to Facility and Mission Requirements for the Nuclear Weapons Complex." Presentation at Los Alamos National Laboratory, Los Alamos. 21 July 1995.

Diamond, Bob. "Concepts of Modeling and Simulation." PC AI Sept.-Oct. 1996: 22-26.

Floss, Peter. Questionnaire response for ReThink. 26 June 1998.

Goble, John, and Brian Wood. "MODSIM III: A Tutorial with Advances in Database Access and HLA Support." Proceedings of the 1998 Winter Simulation Conference. Ed. D. J. Medeiros, Edward F. Watson, John S. Carson, and Mani S. Manivannan. Vol. 1. San Diego: Society for Computer Simulation, 1998. 199-204.

Hammer, Scott. Questionnaire response for ALPHA/Sim. 23 June 1998.

Harrell, Charles. Questionnaire response for ProModel. 5 June 1998.

Harrell, Charles R., and Kerim Tumay. Simulation Made Easy. Norcross: Institute of Industrial Engineers, 1995.

Helm, Terry, Stephen Boerigter, and Stephen Eisenhawer. Stockpile Life-Extension Modeling Presentation at Los Alamos National Laboratory, Los Alamos. 12 Dec. 1997.

Hench, Karen W., J. David Olivas, and Paul R. Finch. Computer Modeling for Optimal Placement of Gloveboxes. LA-UR-97-2187. Los Alamos: Los Alamos National Laboratory, 1997.

Imagine That, Inc. Extend User's Manual. San Jose: Imagine That, Inc. 1997.

Imagine That, Inc. Extend+Manufacturing User's Manual. San Jose: Imagine That, Inc. 1997.

Kelton, W. David, Randall P. Sadowski, and Deborah A. Sadowski. Simulation with Arena. Boston: McGraw-Hill, 1998.

Kjeldgaard, Edwin A., et. al. Planning and Scheduling for Agile Manufacturers: The Pantex Process Model. SAND98-0030. Albuquerque: Sandia National Laboratories, 1998.

Law, Averill M., and W. David Kelton. Simulation Modeling and Analysis. 2nd ed. New York: McGraw-Hill, 1991.

Lawrence Livermore National Laboratory. "Keeping the Nuclear Stockpile Safe, Secure, and Reliable." Science & Technology Review Aug. 1996:7-16.

Lawrence Livermore National Laboratory. Through the science-based Stockpile Stewardship and Management Program, we will ensure the continued safety, security, and reliability of the U.S. nuclear stockpile. UCRL-MI-121423-DNT-1 Rev 1. Livermore: Feb. 1996.

Lenz, John. Questionnaire response for MAST. 25 June 1998.

Meyers, Mike. Questionnaire response for SIMPROCESS. 9 July 1998.

Nordgren, Bill. Questionnaire response for Taylor ED. 1 July 1998.

Orca Computer. Web Site. 26 Jan. 1999 <http://www.OrcaComputer.com/>.

O'Reilly, Jean. Questionnaire response for AweSim and FACTOR/AIM. 17 June 1998.

Pantex Plant. Web Site. 22 Jan. 1999 <http://www.Pantex.gov/>.

Parker, Robert Y. Extend Customization – Experiences and Issues. LA-UR-97-5032. Los Alamos: Los Alamos National Laboratory, 1997.

Parker, Robert Y. Questionnaire response for Extend. 16 June 1998.

Rohrer, Matthew. Questionnaire response for AutoMod and AutoSched. 17 June 1998.

Sadowski, Deborah. Questionnaire response for Arena. 28 July 1998.

Siprelle, Andrew J., Richard A. Phelps, and M. Michelle Barnes. "SDI Industry: An Extend-Based Tool for Continuous and High-Speed Manufacturing." Proceedings of the 1998 Winter Simulation Conference. Ed. D. J. Medeiros, Edward F. Watson, John S. Carson, and Mani S. Manivannan. Vol. 1. San Diego: Society for Computer Simulation, 1998. 349-356.

Swain, James J. "Simulation Goes Mainstream." OR/MS Today Oct. 1997. 26 Jan. 1999 <http://lionhrtpub.com/orms/orms-10-97/Simulation-story.html>.

United States. Dept. of Energy. Plutonium: The First 50 Years. DOE/DP-0137. Washington: GPO, 1996.

Waller, Tony. Questionnaire response for WITNESS. 6 Oct. 1998.

(NOTE: Trademarks or registered trademarks used herein are property of their respective owners.)

Los Alamos
NATIONAL LABORATORY
Los Alamos, New Mexico 87545